# Exercise 1: Implementing the Singleton Pattern

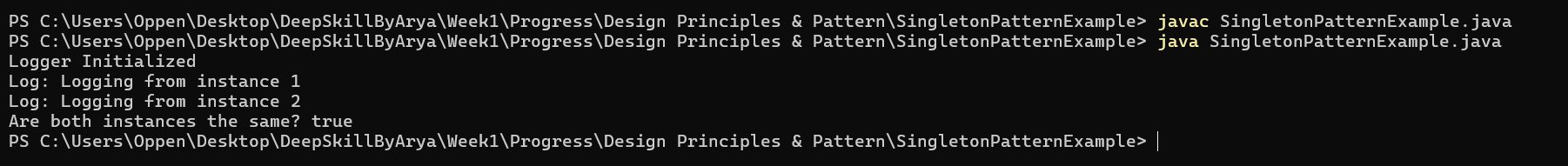
DesignPrinciplesAndPattern

Scenario:  
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

## Code: Logger.java and Main.java (combined as SingletonExample.java)

public class SingletonExample {  
 static class Logger {  
 private static Logger instance;  
 private Logger() {  
 System.out.println("Logger Initialized");  
 }  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger();  
 }  
 return instance;  
 }  
 public void log(String message) {  
 System.out.println("Log: " + message);  
 }  
 }  
 public static void main(String[] args) {  
 Logger logger1 = Logger.getInstance();  
 logger1.log("Logging from instance 1");  
 Logger logger2 = Logger.getInstance();  
 logger2.log("Logging from instance 2");  
 System.out.println("Are both instances the same? " + (logger1 == logger2));  
 }  
}

## Output Screenshot



# Exercise 2: Implementing the Factory Method Pattern

Scenario:  
You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

## Code: FactoryPatternExample.java (All Classes Combined)

public class FactoryPatternExample {  
 public static void main(String[] args) {  
 DocumentFactory wordFactory = new WordDocumentFactory();  
 Document word = wordFactory.createDocument();  
 word.open();  
 DocumentFactory pdfFactory = new PdfDocumentFactory();  
 Document pdf = pdfFactory.createDocument();  
 pdf.open();  
 DocumentFactory excelFactory = new ExcelDocumentFactory();  
 Document excel = excelFactory.createDocument();  
 excel.open();  
 }  
}  
interface Document {  
 void open();  
}  
class WordDocument implements Document {  
 public void open() {  
 System.out.println("Opening Word document.");  
 }  
}  
class PdfDocument implements Document {  
 public void open() {  
 System.out.println("Opening PDF document.");  
 }  
}  
class ExcelDocument implements Document {  
 public void open() {  
 System.out.println("Opening Excel document.");  
 }  
}  
abstract class DocumentFactory {  
 public abstract Document createDocument();  
}  
class WordDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new WordDocument();  
 }  
}  
class PdfDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new PdfDocument();  
 }  
}  
class ExcelDocumentFactory extends DocumentFactory {  
 public Document createDocument() {  
 return new ExcelDocument();  
 }  
}

## Output Screenshot

